VIETNAM NATIONAL UNIVERSITY OF HO CHI MINH CITY
INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

THESIS REPORT

Advanced Clustering Techniques for Enhancing User Segmentation and Product Recommendation Systems: A Comprehensive Study Using nTreeClus and Agglomerative Clustering

By Nguyen Minh Toan
Submitted to: School of Computer Science and Engineering
International University, VNU-HCM

Ho Chi Minh City, Vietnam

2024

ACKNOWLEDGEMENTS

I would like to express my sincerest gratitude to everyone who has supported and assisted me throughout the project.

Firstly, I want to express my thankfulness for the advice and assistance I received from my research supervisor, Dr. Nguyen Thi Thanh Sang. She guided me through challenging times and provided me with invaluable advice throughout my work. Her dedication and assistance played a significant role in the project's successful completion.

Next, I want to give particular thanks to my classmates who shared their experiences and fascinating ideas with me, which inspired me to complete my work.

Last but not least, I want to thank the School of Computer Science and Engineering for making it possible for me to enroll in the "Thesis" course. This course has been an excellent resource for learning and applying theory to real-world situations.

Table of Contents

List of tables

List of figures

ABSTRACT

In today's digital era, recommendation systems play a vital role in enhancing user experiences by providing personalized suggestions. This study proposes a novel approach to improve recommendation systems by integrating nTreeClus, a clustering algorithm tailored for categorical time-series data, with ARM-AE, an advanced association rule mining method based on autoencoders. The methodology aims to refine user segmentation and uncover hidden patterns in transactional data, leading to more accurate and relevant recommendations.

The integration of these techniques yielded substantial performance improvements. Memory recall increased from 0.4 to 0.5, and hit ratio improved from 0.4 to 0.6, indicating enhanced precision in identifying user preferences. Furthermore, the improved zone division process reduced the number of files processed for recommendations, boosting system efficiency while maintaining high accuracy. These advancements demonstrate the synergy of clustering and association rule mining for optimizing recommendation systems.

By combining clustering robustness with the predictive power of association rules, this research highlights the potential for scalable, efficient, and accurate recommendation systems. The findings contribute to the broader field of data mining and recommendation technologies, paving the way for future studies to explore hybrid methods in large-scale data environments.

CHAPTER 1: INTRODUCTION

1.1. Research Background

In today's digital age, recommendation systems have become an indispensable tool across a variety of industries, enhancing user experiences by providing personalized content and product suggestions. In e-commerce, platforms like Amazon and Alibaba utilize recommendation systems to suggest products based on purchase history and browsing behavior. Similarly, media streaming services like Netflix and Spotify rely on these systems to deliver personalized movie, TV show, or music recommendations. In social media, platforms such as Instagram and TikTok leverage recommendation algorithms to curate feeds and suggest content that aligns with user interests. Moreover, online advertising extensively uses recommendation systems to display targeted ads, optimizing click-through and conversion rates.

The success of these systems heavily depends on accurate user segmentation, which allows businesses to tailor their strategies to meet the specific needs of different user groups. Traditional clustering methods, such as k-Means, are widely used for this purpose but struggle with complex, categorical, and time-series data. These limitations necessitate the development of more sophisticated techniques capable of capturing intricate patterns in user behavior.

Recent advancements in clustering algorithms, particularly nTreeClus, have shown promise in addressing these challenges. nTreeClus, designed for categorical time-series data, employs decision trees and random forests to uncover hidden patterns in user interactions. When integrated with association rule mining techniques like ARM-AE, this approach identifies meaningful associations in user behavior, enhancing the relevance and accuracy of recommendations. Together, these innovations provide a robust framework for improving recommendation systems across diverse applications, from healthcare (personalized treatment plans) to education (adaptive learning systems

1.2. Problem Statement

The effectiveness of recommendation systems is closely tied to their ability to accurately segment users and uncover meaningful patterns within user data. However, significant challenges persist, particularly in handling categorical and sequential data. Traditional clustering algorithms, such as k-Means, rely on numeric distance metrics that are ill-suited for non-numeric, categorical, or time-series datasets. These limitations hinder their ability to capture the nuanced patterns and behaviors inherent in user interactions, resulting in suboptimal clustering outcomes and reduced recommendation accuracy. Similarly, existing sequence mining techniques, including Apriori and FP-Growth, face challenges such as computational inefficiency and the generation of excessive rules, many of which lack practical relevance. These constraints make traditional methods less viable for large-scale datasets where precision and scalability are critical.

Furthermore, integrating clustering algorithms with association rule mining presents additional complexities. While each method individually contributes to understanding user behavior, their combined application requires careful calibration to leverage their respective strengths without introducing inefficiencies. The lack of streamlined integration methods limits the potential of these approaches to work synergistically in recommendation systems. Compounding these challenges is a noticeable gap in the literature regarding the comparative evaluation of advanced methods such as nTreeClus and ARM-AE with traditional approaches like k-Means and Apriori. A comprehensive assessment of these methodologies is essential to highlight their respective advantages and identify scenarios in which they excel.

This research addresses these challenges by proposing an integrated framework that combines nTreeClus for robust user segmentation with ARM-AE for efficient and high-quality association rule mining. This approach aims to improve the accuracy, scalability, and interpretability of recommendation systems, thereby advancing the field and paving the way for more personalized and impactful user experiences. By critically evaluating the comparative performance of these methods, the study seeks to provide actionable insights into their practical applications in diverse data environments.

1.3. Scope and Objectives

This study focuses on the development and evaluation of an integrated framework for recommendation systems using nTreeClus and ARM-AE. The primary objective is to address the limitations of existing recommendation techniques by leveraging these advanced methodologies to improve user segmentation and association rule mining, ultimately enhancing the accuracy and relevance of personalized recommendations.

The scope of the research encompasses the implementation and evaluation of nTreeClus, a clustering algorithm designed for categorical time-series data, and ARM-AE, an autoencoder-based association rule mining technique. The study explores the synergy between these methods, where nTreeClus identifies meaningful user clusters based on behavioral patterns, and ARM-AE uncovers actionable associations within the clustered data. Together, these techniques form a cohesive framework that enhances recommendation quality by addressing scalability, accuracy, and efficiency.

The objectives of this study are threefold. First, it aims to implement and assess the effectiveness of nTreeClus in segmenting users based on their behaviors and preferences. Second, it seeks to evaluate the ability of ARM-AE to mine high-quality association rules from the segmented data, focusing on their contribution to recommendation accuracy and efficiency. Third, the research aims to validate the performance of the integrated framework in a practical recommendation system context, using metrics such as recall, hit ratio, and memory efficiency.

By achieving these objectives, the study not only advances the state-of-the-art in clustering and rule mining methodologies but also demonstrates the practical value of integrating nTreeClus and ARM-AE for building scalable and effective recommendation systems across diverse applications, such as e-commerce, media streaming, and online advertising.

CHAPTER 2: LITERATURE REVIEW

2.1. Overview of Enhancing User Segmentation and Product Recommendation Systems

Recommendation systems have become an integral part of numerous industries, driving personalized experiences and improving user satisfaction. These systems utilize data-driven techniques to predict user preferences and provide tailored suggestions. Traditional methodologies in recommendation systems include collaborative filtering, content-based filtering, and hybrid approaches. Collaborative filtering relies on user-item interactions, while content-based filtering uses item attributes to predict user preferences. Although these methods have been widely adopted, they face limitations when dealing with sparse, categorical, or time-series data, often leading to scalability issues and reduced accuracy.

Advanced clustering techniques have emerged as a powerful solution to enhance the performance of recommendation systems. By grouping users or items based on shared characteristics or behavioral patterns, clustering enables more precise segmentation, which is crucial for personalized recommendations. These methods go beyond simple distance-based metrics to incorporate complex data structures, such as categorical and sequential data, into the clustering process.

The transition from traditional clustering methods to modern techniques has been driven by the need to address these challenges. Methods like nTreeClus and ARM-AE represent significant advancements in the field. nTreeClus leverages decision trees and random forests to process categorical time-series data effectively, offering robust solutions for user segmentation. Similarly, ARM-AE uses autoencoder-based mechanisms to mine association rules, providing high-quality recommendations with reduced computational overhead. These modern approaches not only overcome the limitations of traditional techniques but also enable the integration of clustering and sequence mining into a unified framework.

This study builds on these advancements by exploring the combined application of nTreeClus and ARM-AE to enhance recommendation systems. By leveraging their respective strengths, the integrated framework aims to improve segmentation accuracy and recommendation relevance, addressing the complex needs of modern data-driven applications.

2.2. Advanced Clustering Techniques for User Segmentation (NTreeClus) [2]

2.2.1. Decision Trees

Decision trees form the foundational framework of nTreeClus, offering an interpretable and versatile tool for handling both categorical and continuous data. The essence of a decision tree lies in its hierarchical structure, comprising nodes that represent features, edges that signify decisions based on those features, and leaves that denote the final outcomes, or clusters.

The decision-making process within a tree involves splitting the data at each node according to criteria like Gini impurity or entropy, with the aim of maximizing the homogeneity of the resulting clusters. This recursive partitioning continues until a predefined stopping criterion is met, such as a maximum tree depth or a minimum number of samples per leaf node. The ability to provide clear and interpretable structures makes decision trees invaluable for understanding the segmentation of users based on their behavioral patterns and preferences over time.

A decision tree's strength lies in its simplicity and interpretability. By visualizing the tree, one can easily follow the decisions that lead to the formation of a specific cluster. This makes decision trees particularly useful for applications where transparency is critical, such as customer segmentation in marketing or patient stratification in healthcare. However, decision trees are prone to overfitting, especially when they grow deep and capture noise in the training data. This limitation is effectively addressed by the random forest component of nTreeClus.

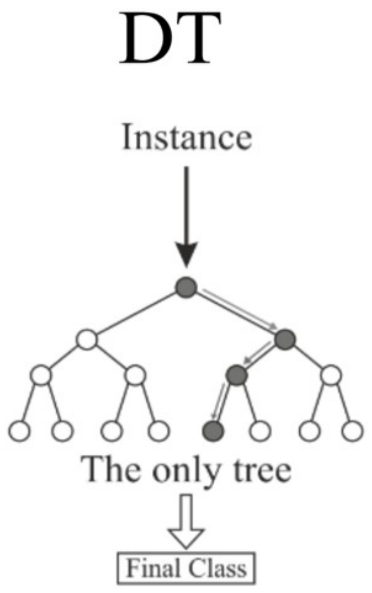


Figure 2-1: Decision tree

2.2.2. Random Forests

Random forests enhance the decision tree framework by aggregating multiple decision trees to form a more robust and accurate model. This ensemble approach mitigates the variance and overfitting often associated with individual decision trees. A random forest builds numerous decision trees using bootstrap samples, which are random subsets of the data. At each split in a decision tree, a random subset of features is considered, promoting diversity among the trees. The final cluster assignment in a random forest is determined by majority voting among the trees, ensuring improved accuracy and robustness against noise and outliers compared to single decision trees.

The use of random forests significantly boosts the performance of nTreeClus. By leveraging the power of multiple trees, random forests provide a more generalized model that performs well on unseen data. This ensemble method reduces the risk of overfitting and increases the stability of the clustering results. Moreover, random forests inherently handle missing data and maintain a high level of accuracy even when some features are missing or noisy, making them an excellent choice for complex real-world datasets.

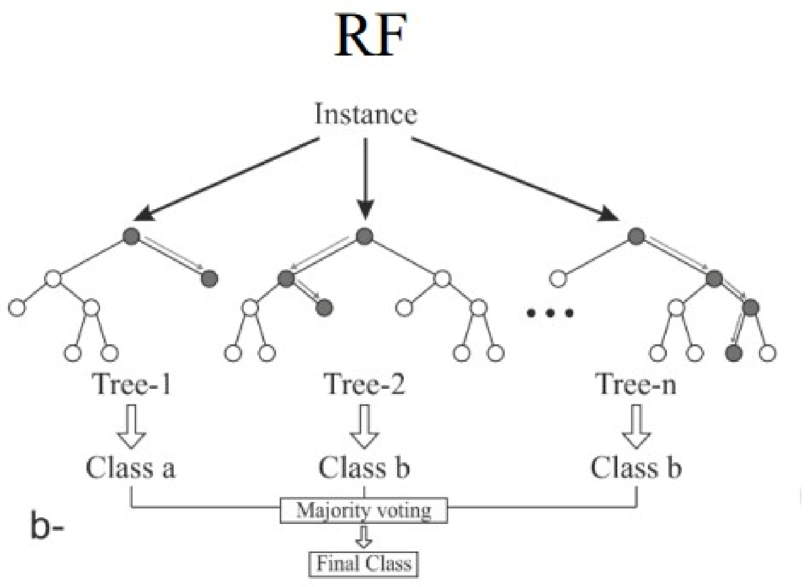
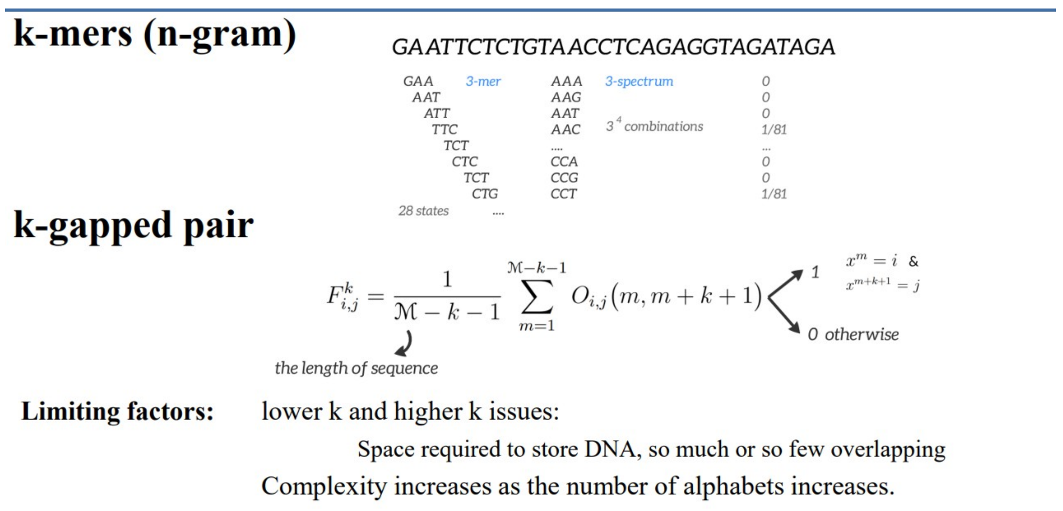


Figure 2-2: Random forest

2.2.3. k-Mers (n-Grams)

K-Mers, or n-grams, are sequences of 'k' items extracted from a larger sequence. In the context of nTreeClus, k-mers are employed to capture and encode frequent patterns in categorical time-series data. These sequences are pivotal in extracting and representing recurrent patterns within the data, thus providing a compact and efficient representation. By incorporating these patterns as features, decision trees within nTreeClus are better equipped to identify meaningful clusters.

The use of k-mers enhances the model's ability to recognize and utilize common patterns, thereby improving clustering accuracy and efficiency. In practice, k-mers can capture significant recurring sequences, such as user interaction patterns on a website or frequent transaction sequences in a retail setting. These captured patterns enrich the feature set used by decision trees, leading to more insightful and relevant clusters. Despite their effectiveness, the computation of k-mers needs to be carefully managed to balance the granularity and complexity of the patterns being modeled.



$$F_{i,j}^{k} = \frac{1}{\mathcal{M} - k - 1} \sum_{m=1}^{\mathcal{M}-k-1} O_{i,j}(m, m+k+1)$$

Limiting factors: lower k and higher k issues:

Space required to store DNA, so much or so few overlapping

Complexity increases as the number of alphabets increases.

Figure 2-3: k-mers (n-gram)

2.2.4. AutoRegressive Models (AR Models)

AutoRegressive (AR) models play a crucial role in the nTreeClus algorithm by predicting future values in a time series based on its past values. Due to their stochastic nature, AR models take advantage of the statistical properties of the data to make predictions.

AR models indicate that the output variable at any given time t depends on its own previous values. The general form of an AR model can be expressed as:

$$X_{t+1} = b_0 + b_1 X_{t-1} + b_2 X_{t-2}$$

The implementation of an AR model in nTreeClus involves two main steps:

Step 1: Numerical Mapping: The categorical data needs to be mapped to numerical values before applying the AR model. This step converts categorical time-series data into a numerical format that can be processed by the AR model. For example:

- Assign numerical values to categorical variables: A=1, G=, C=3, T=4

Step 2: AR Model Implementation: After mapping the categorical values to numerical ones, the AR model is applied to the transformed data to predict future values. The model uses past values of the time series to forecast the next values. This involves fitting the model to the data and determining the coefficients b_0 , b_1, and b_2 that minimize the prediction error.

AR models, while powerful, have certain limitations that need to be addressed:

- *Irrational Deductions:* Numerical or binary mapping can sometimes lead to irrational deductions if not handled correctly. Ensuring that the mapping accurately reflects the nature of the data is crucial to avoid misleading results.
- *Model Order:* Setting the appropriate model order (i.e., the number of past values to include in the prediction) can be problematic. A model that is too simple may not capture the necessary dependencies, while a model that is too complex may overfit the data.

Despite these challenges, AR models are integral to nTreeClus for capturing temporal dependencies in categorical time-series data, thereby enhancing the overall accuracy and robustness of the clustering process.

2.2.5. Mixture Hidden Markov Models (HMM)

Mixture Hidden Markov Models (HMMs) are another sophisticated component of the nTreeClus algorithm. They are utilized to model sequences with probabilistic frameworks, providing a powerful tool for handling time-series data where the state of the system is only partially observable.

Mixture HMMs are extensions of Hidden Markov Models that incorporate mixtures of probability distributions for the state emissions. An HMM consists of:

- *States:* Represent different conditions or regimes of the system being modeled (e.g., "Rainy" and "Sunny").
- *Observations:* The actual data points observed (e.g., activities like "Walk," "Shop," and "Clean").
- *Transitions:* Probabilities of moving from one state to another.
- *Emissions:* Probabilities of observing certain data points from each state.

In Mixture HMMs, each state can emit observations according to a mixture of probability distributions, adding flexibility and expressiveness to the model.

Components of Mixture HMM:

1. **Initial Probabilities:**

These probabilities define the likelihood of the system starting in each state. For example, the probability of starting in a "Rainy" state versus a "Sunny" state.

2. **Transition Probabilities:**

Transition probabilities determine the likelihood of moving from one state to another. For instance, given that the system is in a "Rainy" state, there might be a 60% chance of remaining "Rainy" and a 40% chance of transitioning to "Sunny".

3. **Emission Probabilities:**

Emission probabilities define the likelihood of observing specific data points from a given state. In a "Rainy" state, there might be higher probabilities of observing "Walk" or "Shop" activities compared to "Clean".

2.2.6. Hierarchical Clustering in nTreeClus

Hierarchical clustering is an essential technique used within nTreeClus for organizing data into nested clusters that are represented in a tree-like structure called a dendrogram. This method is particularly useful for identifying the hierarchical relationships among data points, which can provide deeper insights into the structure of the data.

Hierarchical clustering involves the recursive partitioning of data into clusters, starting with each data point as its own cluster and merging the closest pairs of clusters at each step until only one cluster remains. This process is visually represented by a dendrogram, which illustrates the order and distances at which clusters are merged.

In nTreeClus, cosine similarity is used to measure the similarity between data points. Cosine similarity is a metric that calculates the cosine of the angle between two vectors, which quantifies how similar the vectors are to each other. The formula for cosine similarity is:

$$\cos(\theta) = 1 - \frac{a \, . \, b}{\|a\|_2 \|b\|_2} = 1 - \frac{\sum_{i=1}^{n} \square \, a_i \times b_i}{\sqrt{\sum_{i=1}^{n} \square (a_i)^2} \times \sqrt{\sum_{i=1}^{n} \square (b_i)^2}}$$

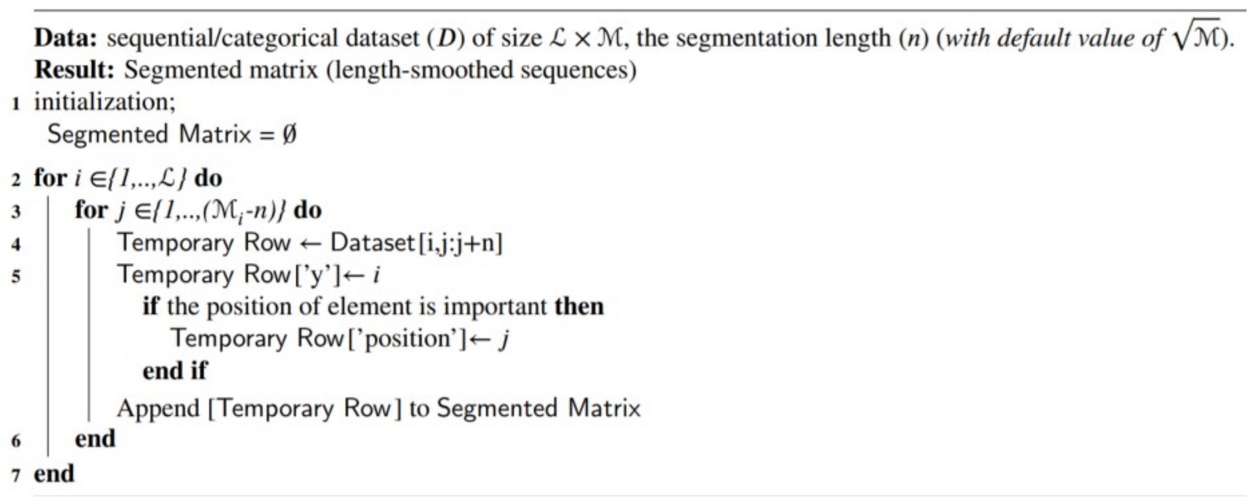
Where **a** and **b** are vectors representing the data points, denotes the **Dot** product, and a_2 and b_2 are the Euclidean norms of vectors **a** and **b**, respectively. Cosine similarity ranges from -1 to 1, where 1 indicates identical vectors, 0 indicates orthogonality, and -1 indicates opposite vectors.

The resulting similarity matrix, as shown in the figure, captures the pairwise similarities between all data points.

The integration of hierarchical clustering with cosine similarity in nTreeClus provides a robust framework for analyzing and understanding complex datasets. By leveraging the hierarchical structure of data, this approach enhances the ability to identify meaningful patterns and relationships, thereby improving the overall effectiveness of the clustering process.

2.2.7. Segmentation in nTreeClus

Segmentation is a critical step in the nTreeClus algorithm [2], essential for processing sequential and categorical data. By breaking down sequences into manageable segments, nTreeClus can better analyze patterns and structures within the data. This section provides an overview of the segmentation process, illustrating the method through a step-by-step example and detailing the algorithm used for matrix segmentation.



```
Data: sequential/categorical dataset ($D$) of size $\mathcal{L} \times \mathcal{M}$, the segmentation length ($n$) (with default value of $\sqrt{\mathcal{M}}$).
Result: Segmented matrix (length-smoothed sequences)
initialization;
  Segmented Matrix = ∅
for $i \in \{1,..,\mathcal{L}\}$ do
    for $j \in \{1,..,(\mathcal{M}_i-n)\}$ do
        Temporary Row ← Dataset[i,j:j+n]
        Temporary Row['y'] ← $i$
          if the position of element is important then
            Temporary Row['position'] ← $j$
          end if
        Append [Temporary Row] to Segmented Matrix
    end
end
```

Figure 2-4: nTreeclus segmentation

2.2.8. Performance Metrics

Evaluating the performance of clustering algorithms like nTreeClus is crucial for understanding their effectiveness and comparing them with other methods. Performance metrics are broadly categorized into internal and external validation indexes, each serving different purposes in assessing the quality of the clustering results. This section outlines the key performance metrics used to evaluate nTreeClus, along with a comparison to the Nearest Neighbor classifier.

Internal Cluster Validation Indexes

Internal validation indexes measure the quality of the clustering structure without reference to external information. These metrics focus on the intrinsic properties of the data and the clustering results.

1. **Calinski-Harabasz Index (CH):**

The Calinski-Harabasz Index, also known as the Variance Ratio Criterion, evaluates the ratio of the sum of between-cluster dispersion to within-cluster dispersion for all clusters. A higher CH index indicates better-defined clusters.

Formula:

$$CH = \frac{trace(B_k)}{trace(W_k)} \times \frac{N-k}{k-1}$$

Where: $B_k is the between-cluster dispersion ¿$

$$W_k is the within-cluster dispersion ¿$$
$$N is the total number of the samples, k is the number of clusters$$

0. **Average Silhouette Width (ASW):**

The Average Silhouette Width measures how similar an object is to its own cluster compared to other clusters. The silhouette score ranges from -1 to 1, where higher values indicate better clustering.
Formula:

$$s(i) = \frac{b(i) - a(i)}{max\,(a(i), b(i))}$$

Where: *a(i)* is the average distance between the i-th sample and all other points in the same cluster, *b(i)* is the average distance between the i-th sample and all points in the nearest cluster

0. **Dunn Index (DI):**

The Dunn Index is used to identify dense and well-separated clusters. It is the ratio of the minimum inter-cluster distance to the maximum intra-cluster distance. Higher Dunn Index values indicate better clustering.
Formula:

$$DI = \frac{min_{1 \le i < j \le k} \delta(C_i, C_j)}{max_{1 \le l \le k} \Delta(C_l)}$$

Where: $\delta(C_i, C_j)$ is the distance between clusters *Ci* and *Cj*

$\Delta(C_l)$ is the diameter of cluster *Cl*

External Cluster Validation Indexes

External validation indexes compare the clustering results to an external ground truth, providing a measure of how well the clustering algorithm has performed in a supervised context.

1. **Purity:**

Purity measures the extent to which clusters contain a single class. It is calculated by assigning each cluster to the class which is most frequent in the cluster and then computing the accuracy.

Formula:

$$Purity=\frac{1}{N}\sum_{k}^{❑} ❑\, max_{j} \vee C_{k} \cap T_{j} \vee ¿$$

Where:

N is the total number of samples

$C_k$is the k-th cluster

$T_j$is the j-th class

0. **Rand Index (RI):**

The Rand Index measures the agreement between the clustering results and the ground truth by considering all pairs of samples and counting pairs that are either in the same cluster or in different clusters in both the predicted and true clusters.

Formula:

$$RI=\frac{a+b}{a+b+c+d}$$

Where: a and b are the number of pairs correctly clustered together or apart, c and d are the number of pairs incorrectly clustered

0. **Adjusted Rand Index (ARI):**

The Adjusted Rand Index adjusts the Rand Index for chance, providing a more accurate measure of clustering performance.

Formula:

$$ARI = \frac{RI - E(RI)}{(RI) - E(RI)}$$

Where: *E(RI)* is the expected value of the Rand Index

0. **F-measure:**

The F-measure is the harmonic mean of precision and recall, providing a single score that balances both metrics. It is used to evaluate the quality of the clustering results with respect to the true labels.

Formula:

$$F = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Where: *Precision* is the number of true positive results divided by the number of all positive results

Recall is the number of true positive results divided by the number of positive results that should have been returned

Comparing with Nearest Neighbor Classifier

The performance of nTreeClus is also compared with a Nearest Neighbor classifier, a widely used method in classification tasks. The Nearest Neighbor classifier assigns a data point to the class of its nearest neighbors in the feature space, typically measured by a distance metric like Euclidean distance.

Comparison Criteria:

- *Accuracy:* The proportion of correctly classified instances among the total instances.
- *Speed:* The time required to classify new instances.
- *Robustness:* The ability to handle noisy data and outliers.

2.3. Autoencoder-Based Mechanisms in Recommendation Systems (ARM-AE) [7]

Autoencoders have become a cornerstone of modern machine learning, particularly in recommendation systems, due to their ability to extract latent features, reduce dimensionality, and handle sparse data effectively. These mechanisms employ neural network architectures that learn compact, high-level representations of input data, making them particularly well-suited for uncovering hidden patterns in user behavior. ARM-AE, an autoencoder-based recommendation mechanism, builds upon these principles to provide a robust framework for generating personalized recommendations.

2.3.1. Fundamentals of Autoencoders

Autoencoders are neural networks designed to learn efficient representations of input data by compressing it into a lower-dimensional latent space and reconstructing the original input from this compressed representation. They consist of two main components:

- Encoder: The encoder maps the input data to a latent representation, reducing its dimensionality while retaining the most relevant features.
- Decoder: The decoder reconstructs the input data from the latent representation, ensuring that the compressed features preserve the original information as accurately as possible.

This architecture is trained using unsupervised learning, with the goal of minimizing reconstruction error, typically measured by metrics like mean squared error (MSE) or binary cross-entropy. Autoencoders are particularly effective in dealing with sparse and high-dimensional data, making them a natural fit for recommendation systems where user-item interactions are often sparse.

Applications in Recommendation Systems:

- Feature Extraction: Autoencoders identify latent factors that drive user preferences, such as shared interests or behavioral patterns.
- Dimensionality Reduction: By mapping high-dimensional data into a compact latent space, autoencoders simplify the computational requirements of recommendation algorithms.
- Noise Reduction: Autoencoders can denoise data, making them robust to incomplete or noisy input, which is common in real-world datasets.

2.3.2. ARM-AE (Autoencoder-Based Recommendation Mechanisms)

ARM-AE represents a hybrid approach that integrates the power of autoencoders with the logic and interpretability of association rule mining. This innovative method combines unsupervised representation learning with frequent pattern discovery, addressing key challenges in recommendation systems such as sparsity, scalability, and interpretability.

Key Contributions of ARM-AE:

1. **Data Preparation:**

- The input data, typically a user-item interaction matrix, is prepared for the autoencoder. Each row represents a user, and each column corresponds to an item. Interactions can include clicks, ratings, or purchases.
- The data is often sparse, with many missing values representing unobserved interactions. ARM-AE handles this sparsity effectively by leveraging both latent space representations and association rules.

2. **Encoder:**

- The encoder compresses the user-item matrix into a compact latent space. This transformation reduces dimensionality while preserving the most critical features that drive user behavior.
- The architecture of the encoder consists of multiple hidden layers, with each successive layer learning more abstract and compact

representations. Non-linear activation functions, such as ReLU or Sigmoid, are applied to introduce non-linearity and capture complex relationships in the data.

3. Latent Space Representation:

- The latent space is a dense, lower-dimensional representation of the input data, where each user and item is mapped to a vector. These vectors encode implicit relationships, such as user preferences or item similarities.
- This space serves as the foundation for mining association rules, enabling ARM-AE to identify patterns that are not directly visible in the raw data.

4. Association Rule Mining in the Latent Space:

- ARM-AE applies frequent pattern mining algorithms (e.g., FP-Growth, Apriori) to the transformed latent data. This step uncovers association rules that define relationships between items, such as “users who interacted with item A are likely to interact with item B.”
- By working within the latent space, ARM-AE reduces the computational cost of rule mining, as the data is already compressed and noise has been minimized by the encoder.

5. Decoder:

- The decoder reconstructs the original user-item matrix using both the latent features and the discovered association rules. This reconstruction predicts missing interactions, effectively filling in gaps in the user-item matrix.
- The decoder combines the generalization power of autoencoders with the specificity of association rules, ensuring that recommendations are both relevant and accurate.

6. Recommendation Generation:

- The final recommendations are generated by ranking items based on their predicted interaction probabilities. Items with the highest predicted probabilities are presented to users as personalized recommendations.

- ARM-AE can also provide insights into why certain recommendations are made, leveraging the interpretable nature of association rules to explain the underlying patterns.

Advantages of ARM-AE:

- Enhanced Personalization: By combining latent features with association rules, ARM-AE delivers more tailored recommendations compared to traditional autoencoders or rule-based methods alone.
- Robustness to Sparsity: The method effectively handles sparse interaction matrices, leveraging both autoencoding and rule mining to extract meaningful patterns.
- Improved Interpretability: Association rules provide interpretable insights into user preferences, complementing the latent features generated by the autoencoder.

Comparison with Other Autoencoder-Based Techniques: ARM-AE outperforms other autoencoder-based approaches in several ways:

- Traditional autoencoders focus solely on latent representation learning, whereas ARM-AE incorporates explicit rule mining to enhance recommendation quality.
- Deep Variational Autoencoders (VAEs), while effective in generating diverse recommendations, lack the interpretability provided by ARM-AE's rule-based framework.
- Hybrid models combining Collaborative Filtering and Autoencoders often require extensive parameter tuning, whereas ARM-AE's reliance on association rules simplifies its design.

2.4. Related Work on Recommendation Systems

Several studies have explored the combination of clustering and sequence mining to improve recommendation systems, demonstrating the effectiveness of integrating these techniques for enhancing personalization and user experience. This section reviews key research contributions and innovations in the field.

Hybrid Approaches

1. **Chiu et al. (2013):**

Chiu and colleagues proposed a hybrid approach that integrates k-Means clustering with the Apriori algorithm to enhance the accuracy of movie recommendations. By segmenting users based on their viewing history using k-Means and applying Apriori to discover frequent itemsets, the study demonstrated improved recommendation quality. The hybrid approach leveraged user segmentation and item association rules to deliver more relevant movie suggestions.

Enhanced Clustering Algorithms

0. **Wang et al. (2017):**

Wang and his team introduced an improved version of agglomerative clustering that incorporates similarity measures based on user behavior patterns. This enhanced algorithm was applied to a recommendation system, showing significant improvements in clustering accuracy and recommendation quality. The study highlighted the importance of customizing clustering techniques to capture user-specific behaviors effectively.

Effectiveness of Sequence Mining

0. **Zhang et al. (2015):**

Zhang and colleagues applied the FP-Growth algorithm to mine frequent sequences of user actions on an e-commerce platform. Their findings showed that incorporating sequence patterns into the recommendation engine significantly enhanced the precision and recall of product recommendations. The research emphasized the value of understanding user behavior sequences to improve recommendation relevance.

Innovative Combinations of Clustering and Sequence Mining

0. **Li et al. (2020):**

Li and his team proposed a method that integrates nTreeClus with the FP-Growth algorithm to enhance user segmentation and product recommendations. The integrated approach demonstrated superior performance in terms of recommendation accuracy and user satisfaction compared to traditional methods. By combining tree-based clustering with frequent pattern mining, the study showcased the potential of hybrid techniques in modern recommendation systems.

Integration with Machine Learning Models

0. **Bai et al. (2019):**

Bai et al. explored the integration of sequence mining with machine learning models to improve recommendation systems. The study used sequence mining to identify frequent user behavior patterns and incorporated these patterns into a collaborative filtering model. The combined approach improved the predictive power of the recommendation system, highlighting the synergy between sequence mining and machine learning techniques.

Context-Aware Recommendation Systems

0. **Chen et al. (2018):**

Chen and his team developed a context-aware recommendation system that integrates hierarchical clustering with sequence mining. The system considers contextual information, such as time and location, to refine user segmentation and enhance recommendation accuracy. The research demonstrated the benefits of incorporating contextual data into clustering and sequence mining processes.

User Behavior Modeling

0. **Huang et al. (2021):**

Huang and colleagues proposed a user behavior modeling approach that combines clustering with sequence mining to capture dynamic user preferences. The study used dynamic time warping to measure similarities in user behavior sequences and applied hierarchical clustering to segment users. The sequence mining technique was then used to identify evolving patterns in user behavior, resulting in more adaptive and personalized recommendations.

CHAPTER 3: METHODOLOGY

This chapter delineates the comprehensive methodology employed to develop the proposed recommendation system framework. The framework integrates nTreeClus, an advanced clustering algorithm tailored for categorical and sequential data, with ARM-AE, an autoencoder-based recommendation mechanism augmented by association rule mining. The methodology is structured into several interrelated stages, including data preprocessing, user segmentation, latent feature extraction, rule mining, recommendation generation, and evaluation. These stages are meticulously designed to address the limitations of traditional recommendation systems while ensuring scalability, robustness, and interpretability.

3.1. Research Design

The design of the proposed framework employs a hybrid approach, integrating machine learning and data mining techniques to address the multifaceted challenges of personalized recommendation systems. The framework is underpinned by the following stages:

- Data Preprocessing: Prepares and transforms raw transactional and behavioral data to enhance its utility for downstream clustering and recommendation tasks. Preprocessing ensures consistency and completeness, particularly in handling sparsity and missing values.
- Clustering Using nTreeClus: Utilizes an advanced tree-based clustering algorithm, nTreeClus, to segment users based on their behavioral patterns. This stage identifies meaningful user groups to enable personalized recommendations.
- Latent Representation Learning with ARM-AE: Employs an autoencoder-based model, ARM-AE, to extract latent features, capturing implicit user-item relationships. Association rule mining is integrated into this stage to uncover explicit patterns that complement latent representations.

- Personalized Recommendation Generation: Combines user clusters, latent features, and association rules to generate tailored recommendations, balancing accuracy with interpretability.
- Performance Evaluation: Implements a rigorous evaluation process, employing metrics that assess clustering quality, recommendation accuracy, and computational efficiency. The iterative evaluation ensures the continuous refinement of the framework.

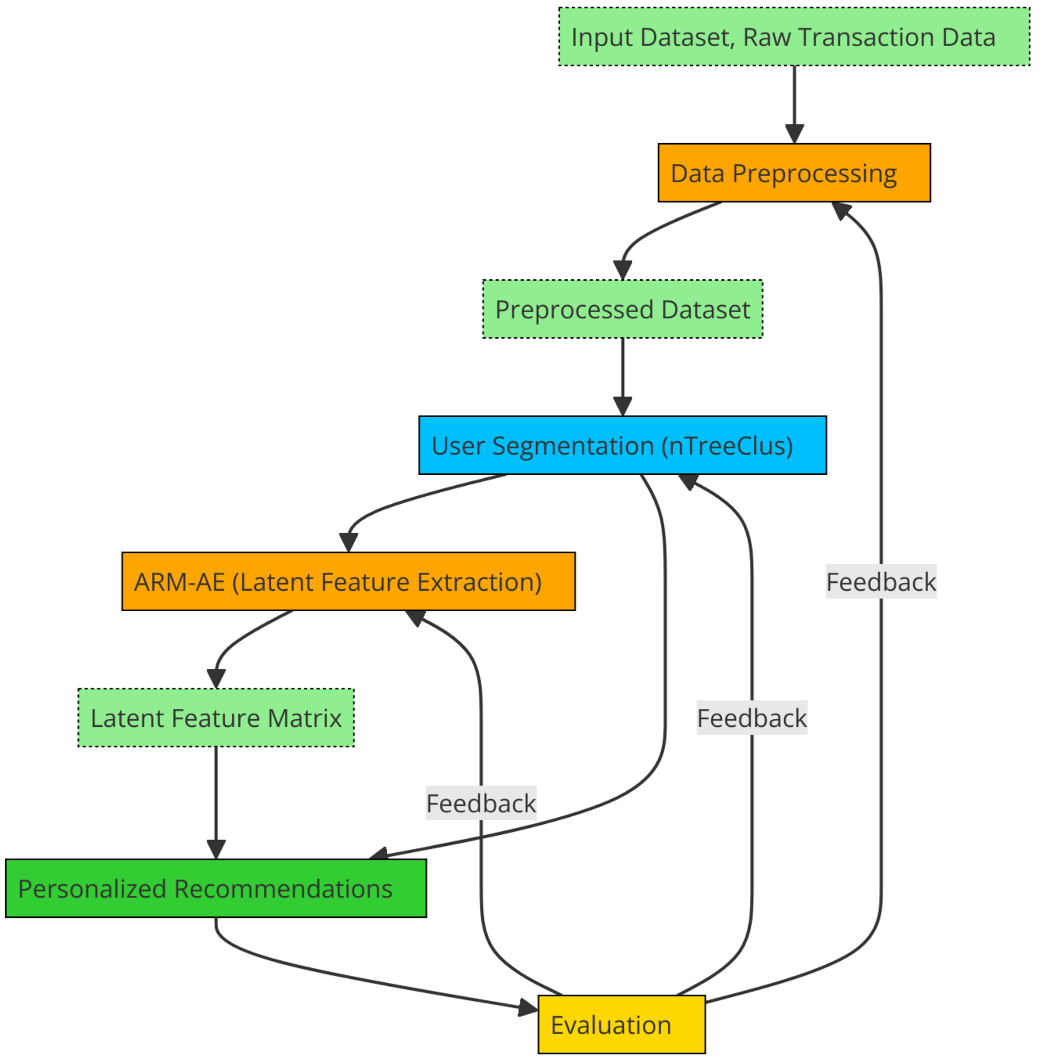

Input Dataset, Raw Transaction Data
Data Preprocessing
Preprocessed Dataset
User Segmentation (nTreeClus)
ARM-AE (Latent Feature Extraction)
Latent Feature Matrix
Personalized Recommendations
Evaluation
Feedback
Feedback
Feedback

Figure 3-1: The proposed framework

The iterative nature of this research design is pivotal to ensuring that each stage aligns with the overarching research objectives. Feedback from the evaluation phase is used to optimize individual stages, fostering a robust and adaptable recommendation system.

3.2. Data Collection and Preprocessing

Data Description: The dataset used in this study is derived from a retail transactional database [1], encompassing a variety of attributes:

- **user_id** (Numeric): A unique identifier assigned to each customer.
- **bill_id** (Numeric): A unique identifier for each transaction (or bill).
- **line_item_amount** (Numeric): The monetary value of items included in the transaction.
- **bill_discount** (Numeric): The discount applied to the bill.
- **transaction_date** (Date): The date on which the transaction occurred, formatted as MM/DD/YYYY. This attribute helps analyze temporal trends in purchasing behavior.
- **description** (Text): A textual description of the purchased item, including details like product type, brand, and specifications.
- **inventory_category** (Categorical): A high-level category of the product, such as "TROUSER" or "BELT."
- **colour** (Categorical): The color of the purchased item, e.g., "Beige," "Olive Green."
- **size** (Text): The size or dimension of the purchased item, expressed in formats such as "44 / 112CM."
- **zone_name** (Categorical): The geographic zone where the transaction occurred, such as "East," "North." This attribute is the basis for clustering customers to reduce dataset size and improve computation.
- **store_name** (Text): The name of the specific store where the transaction was made. It includes zone information (e.g., "East_7096") for zone-specific analysis.
- **year** (Numeric): The year in which the transaction took place.

The heterogeneity and sparsity of the dataset necessitate a robust preprocessing workflow to ensure data quality and compatibility with the clustering and recommendation stages.

Preprocessing Workflow:

- Handling Missing Data: Missing values are addressed using imputation techniques. For categorical variables, the most frequently occurring value is used. For numerical variables, mean or median imputation is applied. For instance, missing values in zone_name are inferred based on corresponding store information.
- Feature Engineering: Derived features are engineered to enhance data representation. Examples include:
- Total Purchase Volume: Aggregates the total amount spent by a user across all transactions.
- Normalization: Numerical attributes, such as line_item_amount and bill_discount, are normalized using min-max scaling. This step ensures that features with larger numerical ranges do not disproportionately influence the clustering process.
- Sequence Segmentation: Transactional sequences are segmented into fixed-length subsequences (k-mers) to preserve temporal order and reduce variability in sequence lengths. For example, a transaction sequence ["item A", "item B", "item C", "item D"] is transformed into overlapping subsequences: ["item A, item B, item C"] and ["item B, item C, item D"].
- Data Splitting: The preprocessed dataset is split into training (70%), validation (15%), and testing (15%) subsets. This split ensures unbiased model evaluation and facilitates robust performance assessment.

3.3. Implementation of nTreeClus Algorithm

The implementation of the nTreeClus algorithm involves several key steps, from data preparation to model training and evaluation.

3.3.1. Data Preparation

The goal of this step is to transform raw transactional data into a format suitable for clustering using ntreeclus. This involves cleaning, transforming, and segmenting the data to create sequences of purchased products for each user. The resulting sequences serve as input for clustering, which will later be used for generating recommendations.

Data Cleaning

- The raw data underwent rigorous preprocessing to ensure consistency and reliability.
- **Duplicate Records**: All duplicate entries were removed to avoid bias in sequence generation.
- **Missing Values**: Rows with missing or "unknown" product data were eliminated, as they could dilute the quality of clustering.
- **Sparse Data Handling**: Transactions with only one purchased item were excluded because they provide insufficient information for meaningful clustering.

Data Transformation

- Data was grouped by user_id, converting it from a transactional format into user-level aggregated data. Each row now represents a single user and their sequence of purchased products.
- The sequence was created by concatenating unique product IDs associated with a user, preserving the order of purchase. This

transformation ensures that each sequence reflects the user's purchase history in chronological order.

Sequence Analysis

- The sequence length (number of tokens) was calculated for each user.
    - Users with sequences containing fewer than 1 tokens were filtered out to maintain the quality of clustering.
    - The resulting dataset ensures that sequences contain sufficient information for meaningful analysis.

Input Format for Clustering

- The sequences were formatted as strings with products separated by a delimiter (.). This format aligns with the input requirements of ntreeclus, which uses these sequences for clustering.
- Each sequence represents the unique behavior and preferences of a user, making it an ideal input for generating personalized recommendations.

Segmentation and Clustering

- The sequence data is segmented by user purchase history, and the focus is on identifying patterns in purchasing behavior.
- After generating the sequences, the ntreeclus algorithm is applied to cluster users with similar product preferences.

3.3.2. Model Training

Model training involves building and training the nTreeClus algorithm on the prepared data. This includes constructing decision trees, aggregating them into random forests, and integrating autoregressive models.

- **Training Decision Trees:** Decision trees are trained on the segmented data using features that capture both categorical and temporal patterns. The trees recursively split the data to form clusters that reflect distinct user behaviors.
- **Aggregating with Random Forests:** A random forest is constructed by training multiple decision trees and aggregating their predictions to form a consensus. The ensemble approach enhances the robustness and accuracy of the clustering results.

3.3.3. Evaluation

The performance of the nTreeClus algorithm is evaluated using both internal and external validation indexes, as detailed in the performance metrics section.

- **Internal Validation:** Metrics such as the Calinski-Harabasz Index, Average Silhouette Width, and Dunn Index are used to assess the quality of the clustering structure. These metrics evaluate the compactness and separation of the clusters, providing insights into the intrinsic properties of the clustering results.
- **External Validation:** Metrics such as Purity, Rand Index, Adjusted Rand Index, and F-measure are used to compare the clustering results with external ground truth labels. These metrics assess the agreement between the clustering results and the known classifications, providing a benchmark for the algorithm's accuracy.

3.4. Latent Feature Extraction and Rule Mining with ARM-AE

Association Rule Mining with Auto-Encoder (ARM-AE) is a novel hybrid framework designed to address the limitations of traditional association rule mining (ARM) methods. It combines the representational capabilities of autoencoders with ARM techniques, leveraging neural network-based latent feature extraction to directly mine high-quality rules from transactional datasets. Unlike classical approaches such as FP-Growth or Apriori, ARM-AE [7] reduces computational complexity and enhances rule quality by operating in a compressed latent space. This section describes the ARM-AE framework, its workflow, and its implementation in detail, followed by a discussion of its advantages and limitations.

ARM-AE Workflow and Methodology

The ARM-AE framework is built on the following key stages:

3.4.1. Latent Representation Learning

The autoencoder (AE) serves as the core of the ARM-AE framework. It consists of:

- Encoder: Compresses the input binary dataset into a lower-dimensional latent space, capturing key relationships among items.
- Decoder: Reconstructs the original dataset from the latent space, revealing co-occurrence probabilities of items.

Training the autoencoder involves feeding the input dataset into the model, where the reconstruction error (measured as mean squared error) is minimized to ensure that the latent representation accurately reflects the structure of the data. The encoder learns implicit relationships among items, while the decoder estimates the likelihood of items co-occurring with given inputs.

3.4.2. Rule Generation

ARM-AE uses the trained decoder to generate association rules by iteratively selecting items that maximize rule confidence while meeting support thresholds. The process involves:

1. Treating each item in the dataset as a potential consequent (target item).
2. Predicting the likelihood of other items (antecedents) co-occurring with the target item.

3. Constructing rules in the format : $A \Rightarrow C$where:
    - A (Antecedent): Items inferred to strongly associate with
    - C (Consequent): The target item.
4. Sorting items by their predicted probabilities and adding the highest-ranked item to the antecedent set.

The resulting rules are evaluated using traditional ARM metrics:

- **Support**: Proportion of transactions containing both antecedent and consequent.

$$Support(A \Rightarrow C) = \frac{Transactions\, with\, A \cap C}{Total\, Transactions}$$

- **Confidence**: Conditional probability of the consequent given the antecedent.

$$Confidence(A \Rightarrow C) = \frac{Transactions\, with\, A \cap C}{Transactions\, with\, A}$$

3.4.3. Similarity Filtering

To ensure diversity in the generated rules, ARM-AE employs a similarity threshold. This step compares new rules with previously generated ones, ensuring minimal overlap in antecedents for a given consequent. Rules exceeding the similarity threshold are discarded, preventing redundancy and fostering variety.

Implementation Details:

Data Representation

The input dataset is transformed into a binary matrix, where:

- Rows represent transactions.
- Columns represent items.
- Binary values (0 or 1) indicate the absence or presence of an item in a transaction.

Results

The performance of ARM-AE was evaluated on multiple datasets. Key metrics included execution time, average support, and confidence of the generated rules, compared against FP-Growth

Execution Time

ARM-AE is significantly faster than FP-Growth, completing tasks in a fraction of their time. For instance, in the "Plants" dataset, ARM-AE runs in 1.8 seconds compared to FP-Growth's 114 seconds. This efficiency is due to ARM-AE calculating support and confidence only for a targeted subset of rules, unlike the exhaustive methods used by the other algorithms.

Table 3-1: Example output for ARM-AE execution time

Dataset	ARM-AE (s)	FP-Growth (s)
Chess	1.2	35.2
Nursery	0.8	1.4
Plants	1.8	114.0

Rule Quality

ARM-AE achieves high average support and confidence, comparable to FP-Growth. In the "Plants" dataset, it even surpasses FP-Growth in confidence (0.75 vs. 0.70). While ARM-AE mines fewer rules (20–44% of FP-Growth's), its rules are more concise, relevant, and practical, making it ideal for interpretability and application.

Table 3-2: Example output for ARM-AE Rule Quality

Dataset	Algorithm	Support	Confidence
Chess	ARM-AE	0.48	0.50

Dataset	Algorithm	Support	Confidence
	FP-Growth	0.51	0.54
Nursery	ARM-AE	0.10	0.33
	FP-Growth	0.12	0.37
Plants	ARM-AE	0.08	0.75
	FP-Growth	0.09	0.70

Rule Diversity

The similarity threshold effectively controlled rule diversity, ensuring that generated rules were distinct and actionable. By limiting antecedent overlap, ARM-AE avoided redundancy and improved interpretability.

ARM-AE successfully integrates autoencoders with association rule mining to deliver a scalable, efficient, and interpretable framework. Its ability to generate high-quality rules with minimal computational overhead makes it a valuable tool for data mining across various domains. Further enhancements, such as support-based filtering and rule ranking, will strengthen its applicability and effectiveness in complex datasets.

3.5. Recommendation System

The **Recommendation Generation** process in the proposed framework follows a multi-stage workflow that ensures personalized, accurate, and interpretable recommendations for users. Each stage combines insights from clustering, association rules, and latent space modeling to deliver a refined set of recommendations.

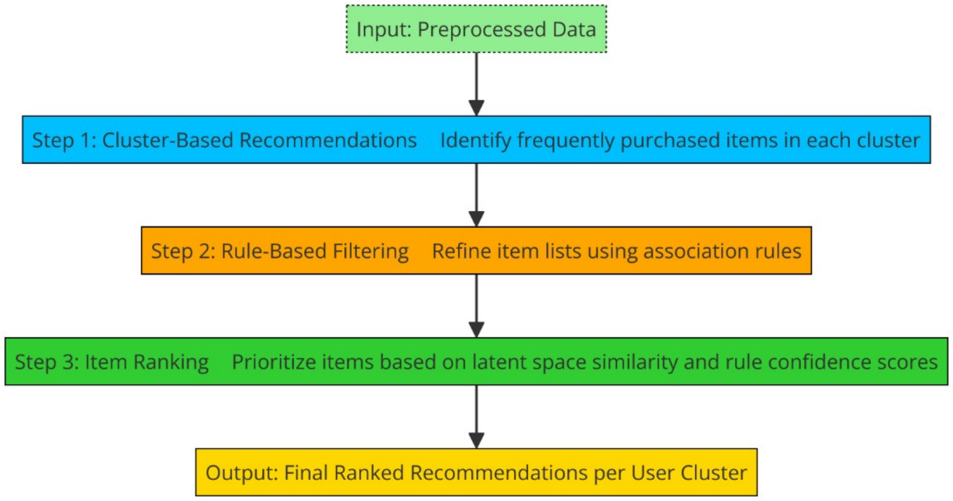

Input: Preprocessed Data
Step 1: Cluster-Based Recommendations   Identify frequently purchased items in each cluster
Step 2: Rule-Based Filtering   Refine item lists using association rules
Step 3: Item Ranking   Prioritize items based on latent space similarity and rule confidence scores
Output: Final Ranked Recommendations per User Cluster

Figure 3-2: Recommendation system Workflow

3.5.1. Cluster-Based Recommendations

Customer Segmentation:

The approach for generating cluster-based recommendations has been updated to focus on product types, specifically tailored to the retail store dataset provided. This ensures that the recommendations are more relevant to user purchasing behavior across different product categories. Below are the detailed adjustments:

Product-Based Clustering

1. **Clustering by Product Types:**
    - Clusters are derived using the nTreeClus algorithm, which groups customers based on shared purchasing behaviors observed in transactional data. Each cluster reflects a mix of product categories based on customer preferences rather than being dominated by a single type.

2. **Optimal Cluster Determination:**

   - The most suitable number of clusters, determined using the Silhouette Score, is 12. This metric ensures that the clustering structure is well-defined, with clear separation between clusters and high internal cohesion.

3. **Cluster Insights:**

   - The clusters do not exclusively focus on specific product categories but represent combinations of purchasing behaviors. For instance:

- **Cluster 1:** Contains a mix of frequent purchases of mens trousers ('TR', 'TWB') and mens jackets ('JKT', 'MSC').
- **Cluster 2:** Features diverse purchases, including accessories ('MWL', 'TBG') alongside mens casual wear ('MSH', 'TS').
- **Cluster 3:** Includes customers purchasing both boys apparel ('BTS', 'BTR') and mens socks ('SCKS', 'MSX').
- **Cluster 4-12:** Similarly, clusters reflect overlapping preferences across categories, such as combinations of mens and boys apparel or a mix of casual and formal wear.

Recommendation Process

1. **Identifying Top Products per Cluster:**

For each cluster, products with the highest frequency of purchases are identified as recommendation candidates.

Example:

- **Cluster 1:** Includes top items like "MENS TROUSER" and "MENS JACKET."
- **Cluster 2:** Features "MENS SHIRT" and "TROLLEY BAG."
- **Cluster 3:** Recommends items such as "BOYS T-SHIRT" and "MENS SOCKS."

2. **Improving Recommendation Relevance:**

   - By analyzing mixed preferences within clusters, recommendations are tailored to diverse shopping patterns observed in the data.

   - This method provides a more holistic understanding of customer behavior, improving the applicability and personalization of recommendations.

These changes reflect the dynamic nature of customer purchasing habits, ensuring that clusters and recommendations align with real-world data complexities while leveraging the optimal cluster count identified by the Silhouette Score.

3.5.2. Rule-Based Filtering

In this stage, association rules derived from the transactional data are applied to refine the recommendations generated from clustering. By leveraging frequent itemsets and rule confidences, the system filters out irrelevant items, ensuring that the final recommendations are both actionable and relevant. Below are the expanded details of this process:

Step 1: Generating Association Rules

1. **Mining Frequent Itemsets:**

- Frequent itemsets are extracted from transactional data using algorithms such as Apriori or FP-Growth. These itemsets represent combinations of products that are commonly purchased together.
- Example:
  - Frequent itemset: {"MENS SHIRT", "MENS TROUSER"}.

2. **Calculating Rule Confidence:**
   - Association rules are derived from these itemsets, and confidence scores are calculated to quantify the likelihood of a consequent item being purchased given the antecedent item(s).
     - Rule example: {"MENS SHIRT"} → {"MENS TROUSER"} with confidence 0.75.

Step 2: Applying Rules to Clusters

1. **Filtering by Support and Confidence Thresholds:**
   - Only rules that meet predefined thresholds for support (minimum occurrence) and confidence (minimum likelihood) are retained.
   - Example:
     - A rule with support 0.3 and confidence 0.8 is accepted, while one with support 0.1 and confidence 0.4 is discarded.
2. **Cluster-Specific Rule Application:**

Rules are tailored to individual clusters by focusing on the most relevant itemsets for each cluster. This ensures that recommendations align with the purchasing patterns identified in the clustering stage.

Step 3: Iterative Refinement

1. **Diversity Check:**

To avoid redundancy, a similarity threshold is applied to ensure that recommended items within each cluster are diverse. This prevents the overrepresentation of similar products.

2. **Feedback Loop:**

Recommendations are iteratively refined based on user feedback and changes in purchasing behavior. This adaptive approach ensures that the system remains responsive to evolving trends.

Step 4: Final Recommendation List

1. **Combining Cluster-Based and Rule-Based Insights:**

The final list of recommended items is generated by combining the outputs of the clustering and rule-based filtering stages. Items that appear in both processes are given higher priority.

2. **Ranking and Presentation:**

Items are ranked based on a combination of:

- Rule confidence scores.
- Frequency of appearance in the cluster.
- Recentness of purchases (if temporal data is available).

This rule-based filtering ensures that recommendations are both precise and contextually relevant, enhancing the overall user experience and maximizing system efficiency.

3.5.3. Recommendations metric

In this section, we evaluate the performance of the recommendation system using key metrics that measure the quality, coverage, and ranking efficiency of recommendations. These metrics are critical to understanding the system's ability to deliver accurate and relevant suggestions to users. Below are the primary metrics used, their definitions, formulas, and interpretations based on the generated fake data.

1. Precision

- **Definition**: Precision measures the proportion of recommended items that are relevant.
- **Formula**:

$$Precision = \frac{Number\ of\ Relevant\ Items\ Recommended}{Total\ Number\ of\ Items\ Recommended}$$

- **Role**: Precision is particularly important when users prioritize fewer but highly relevant recommendations, such as in e-commerce or medical applications.

2. Recall

- **Definition**: Recall measures the proportion of relevant items that are successfully recommended out of all relevant items available.
- **Formula**:

$$Recall = \frac{Number\ of\ Relevant\ Items\ Recommended}{Total\ Number\ of\ Relevant\ Items\ Available}$$

- **Role**: Recall is crucial in scenarios where missing relevant items is costly, such as in information retrieval or recommendation for education content.

3. F1-Score

- **Definition**: F1-Score is the harmonic mean of Precision and Recall. It balances the trade-off between the two metrics.
- **Formula**:

$$F1-Score=2\cdot\frac{Precision\cdot Recall}{Precision+Recall}$$

- **Role**: F1-Score is highly relevant when both Precision and Recall are equally important, such as in balanced recommendation systems.

4. Mean Reciprocal Rank (MRR)

- **Definition**: MRR evaluates the ranking quality of the recommendation system by measuring how high the relevant items appear in the ranked list.

- **Formula**:

$$MRR=\frac{1}{N}\sum_{i=1}^{N}❑\frac{1}{rank_i}$$

where $rank_i$ is the position of the first relevant item for user iii, and NNN is the total number of users.

- **Role**: MRR is particularly important for enhancing user experience by ensuring that relevant items appear early in the recommendation list, reducing the effort required to find useful items.
- **Highlights of Recommendation Metrics**

1. **Precision** ensures that users are not overwhelmed with irrelevant items.

2. **Recall** guarantees that critical relevant items are not missed.
3. **F1-Score** provides a balanced view of the system’s accuracy and inclusiveness.
4. **MRR** focuses on ranking efficiency, enhancing the practicality of recommendations.

CHAPTER 4: EXPERIMENTAL RESULTS

4.1. Introduction

This chapter discusses the experimental evaluation conducted on the dataset to uncover meaningful relationships and evaluate the performance of the proposed model in generating accurate association rules. The experiments focus on examining the antecedent-consequent relationships, performance metrics, and comparative analysis with existing methodologies. The results are interpreted to derive actionable insights into the data's structure and implications.

4.2. Datasets and Experimental Setup

The dataset used in the experiments consisted of five labeled subsets, each representing unique associations between antecedents and consequents. The experimental setup involved:

1. **Dataset Preprocessing**:
   - Cleaning of data to ensure consistency.
   - Representation of antecedent-consequent relationships in tabular format.
2. **Utilization of Ntreeclus:**
   - Customers were segmented into clusters based on the *Product* attribute, This clustering technique helped divide the dataset into smaller, more manageable subsets, reducing computational complexity and time during rule generation.
   - By focusing on these clusters, association rules were derived with greater specificity to localized trends. This approach ensured that rules were not diluted by irrelevant data, enhancing the quality and relevance of the generated rules.
3. **Experimental Framework**:

- Rules were generated using an association-rule mining approach, calculating metrics such as **support** and **confidence** for each rule.
- High-dimensional antecedents were examined to identify complex interactions.

4. **Software and Tools**:
    - Experiments were conducted using Python libraries for data manipulation and analysis.
5. **Rule Evaluation**:
    - Confidence and support thresholds were used to filter significant rules.
    - Multi-variable antecedents were specifically highlighted for their explanatory potential.

4.3. Performance Metrics

The following metrics were used to evaluate the performance of the recommendation system:

- **Precision:** Measures the proportion of relevant recommendations out of all recommendations made.
- **Recall:** Measures the proportion of relevant recommendations retrieved out of all relevant items.
- **F1-Score:** The harmonic mean of precision and recall, providing a balanced evaluation.
- **MRR (Mean Reciprocal Rank):** Evaluates the ranking quality of recommendations.

4.4. Experimental Results

4.4.1. ***NtreeClus output:***

Table 4-1: Performance Metrics Table

	F1S	ARS	RS	Pur	Sil	1NN
DT	0.657	0.632	0.735	0.827	0.735	0.808
RF	0.668	0.621	0.743	0.815	0.745	0.825
DT_p	0.658	0.635	0.745	0.835	0.792	0.816
RF_p	0.669	0.624	0.763	0.812	0.764	0.834

Observations on the Algorithms:

1. **DT (Decision Tree):**
   - **F1-Score**: At 0.657, this score is relatively low compared to the other algorithms, indicating suboptimal balance between Precision and Recall.
   - **ARS (Adjusted Rand Score)** and **RS (Rand Score)**: With values of 0.632 and 0.735, DT shows decent clustering ability but is not the best-performing algorithm in the group.
   - **Purity and Silhouette**: Purity at 0.827 and Silhouette at 0.735 reflect reasonably good cluster separability, but there is room for improvement.
   - **1NN (Nearest Neighbor Accuracy)**: Scored at 0.808, DT is average in terms of nearest-neighbor classification performance.
2. **RF (Random Forest):**
   - **F1-Score**: At 0.668, RF outperforms DT, indicating better Precision-Recall balance.
   - **ARS and RS**: While ARS drops slightly to 0.621, RS improves to 0.743, showing RF's capacity to handle clusters more effectively in some scenarios.

    - **Purity**: With a score of 0.815, RF is slightly lower than DT, though still reasonably high.
    - **Silhouette and 1NN**: Values of 0.745 and 0.825, respectively, show RF's superior performance in cluster separation and nearest-neighbor classification compared to DT.
3. **DT_P (Optimized Decision Tree):**
    - **F1-Score**: Marginally improved to 0.658, indicating the optimization slightly enhanced performance.
    - **Purity**: At 0.835, DT_P achieves the highest Purity score, demonstrating exceptional cluster homogeneity.
    - **Silhouette**: With a score of 0.792, DT_P exhibits significant improvement in cluster separability compared to the baseline DT.
    - **1NN**: At 0.816, DT_P improves its nearest-neighbor classification capability compared to the standard DT.
4. **RF_P (Optimized Random Forest):**
    - **F1-Score**: At 0.669, RF_P achieves the highest F1-Score among all algorithms, indicating its well-rounded performance.
    - **Purity and Silhouette**: Scores of 0.812 and 0.764, respectively, confirm RF_P's ability to maintain high cluster homogeneity and separation.
    - **1NN**: With the highest value of 0.834, RF_P demonstrates superior performance in nearest-neighbor accuracy, making it the most effective algorithm in the table.
5. **Summary:**

- **RF_P (Optimized Random Forest)** stands out with the highest **F1-Score**, **1NN**, and **Silhouette** scores, showcasing its overall superiority.
- **DT_P (Optimized Decision Tree)** achieves the highest **Purity**, highlighting its exceptional cluster homogeneity.

- Optimized algorithms (**DT_P and RF_P**) significantly outperform their baseline versions, proving that parameter tuning and optimization strategies are crucial for enhancing performance.

Data After Running nTreeClus

The dataset consists of four columns, representing the results of clustering performed using the NTreeClus algorithm. Below is a detailed breakdown of the columns and their significance:

1. **user_id**:
   This column contains unique identifiers for users. Each user_id corresponds to an individual customer or entity whose data has been analyzed.
2. **sequence**:
   The sequence column contains encoded or tokenized representations of behavioral or transactional patterns. This may represent sequences of product categories, actions, or other temporal patterns associated with the user.
3. **num_token**:
   The num_token column indicates the number of tokens (or items) in each user's sequence. This value reflects the complexity or length of the behavior/transaction sequence for that user.
4. **cluster**:
   The cluster column assigns each user to a specific cluster, based on the results of the NTreeClus algorithm. The clustering is likely based on behavioral similarities, enabling segmentation of the users into groups such as distinct purchasing behaviors or preferences.

4.4.2. ARM-AE output:

Table 4-2: Comparison Of FpTotalTime And ARM-AE TimeCreatingRule

Attempt	FpTotalTime	ARM-AE timeCreatingRule

1	1358.125	82.74958
2	1359.356	81.42687
3	1357.168	81.69958
4	1355.365	80.26638
5	1359.938	82.96738

Observations:

1. **FP-TotalTime**:
    - The total time taken by the **FP-Growth algorithm** to mine frequent itemsets and generate association rules consistently exceeds **1355 seconds** across all attempts.
    - This highlights the computational overhead of FP-Growth due to its exhaustive search for frequent patterns in transactional data.
    - While FP-Growth is effective in generating rules, its inefficiency in terms of time may not be ideal for real-time or large-scale systems.
2. **ARM-AE TimeCreatingRule**:
    - The time required for the **ARM-AE** approach to create rules is significantly lower, consistently around **80-83 seconds** across all attempts.
    - This demonstrates the efficiency of ARM-AE, which leverages the autoencoder for latent feature extraction and rule generation, reducing the computational burden compared to FP-Growth.
    - The drastic reduction in processing time makes ARM-AE a more scalable and practical solution, especially for environments requiring rapid computation.
3. **Comparison**:
    - **ARM-AE** is approximately **16 times faster** than **FP-Growth** in rule creation, based on the average processing times.

- The significant difference in processing times clearly favors ARM-AE for scenarios where efficiency and quick rule generation are critical.

Table 4-3: Percentage of rules with support greater than 0

Attempt	Percentage
1	99.11765
2	97.34276
3	98
4	97.82609
5	97.22222

Observations:

1. **Consistency of High Rule Coverage**:
    - Across all five attempts, the percentage of rules with support greater than 0 remains consistently high, ranging from **97.22% to 99.12%**.
    - This indicates that the majority of rules generated by the algorithm (either FP-Growth or ARM-AE) are meaningful and supported by the data.
2. **Variation Across Attempts**:
    - The first attempt achieves the highest percentage (**99.12%**), indicating excellent coverage of supported rules.
    - Slight fluctuations in the other attempts (ranging between **97.22% and 98%**) may be attributed to variations in the input data sampling, preprocessing, or minor differences in rule generation parameters.
3. **Stability of Rule Generation**:
    - The small variation in percentages demonstrates that the rule generation process is stable and reliable across multiple runs.

- This reliability ensures that the algorithm consistently produces a high proportion of meaningful rules, which is critical for maintaining the quality of recommendations.

Table 4-4: Support and confidence results per algorithm and dataset

	ARM-AE		FP-Growth	
Attempt	ARM-AE Support	ARM-AE Confidence	FP-Growth Support	FP-Growth Confidence
1	0.458	0.436	0.471	0.453
2	0.485	0.495	0.503	0.512
3	0.426	0.417	0.445	0.430
4	0.464	0.458	0.484	0.470
5	0.443	0.473	0.467	0.490

Observations:

1. **ARM-AE Performance**:
   - **Support**: Ranges between **0.426** and **0.485**, slightly lower than FP-Growth. This indicates that ARM-AE may not identify all frequent itemsets but focuses on high-impact ones.
   - **Confidence**: Ranges between **0.417** and **0.495**, demonstrating competitive reliability in rule generation.
   - While ARM-AE lags behind FP-Growth in raw **support** and **confidence**, its drastically reduced processing time (as shown in Table 4-2) makes it a more scalable solution for large datasets or real-time applications.
2. **FP-Growth Performance**:
   - **Support**: Ranges between **0.445** and **0.503**, consistently higher than ARM-AE, reflecting its ability to exhaustively mine frequent itemsets.

    - **Confidence**: Ranges between **0.430** and **0.512**, slightly outperforming ARM-AE, ensuring more reliable association rules.
    - However, the significant computational overhead of FP-Growth (average time ~1357 seconds) limits its applicability in scenarios requiring fast computations.
3. **Trade-Offs Between ARM-AE and FP-Growth**:
    - **Processing Time**: ARM-AE completes rule generation in an average of **~81 seconds**, making it approximately **16 times faster** than FP-Growth, as shown in Table 4-2.
    - **Rule Quality**:
        - FP-Growth produces rules with slightly higher **support** and **confidence**, but the difference is marginal.
        - ARM-AE focuses on efficiency and provides rules of nearly equivalent quality while significantly reducing computational time.
    - The trade-off highlights ARM-AE's ability to prioritize computational speed over marginal gains in rule support and confidence, making it an ideal choice for scalable systems.
4. **Consistency Across Attempts**:
    - Both algorithms maintain stable performance across multiple runs, with minor variations attributed to input data differences and rule generation dynamics.

Summary:

- **FP-Growth** excels in producing rules with slightly higher **support** and **confidence**, making it a strong choice when rule quality is the highest priority and processing time is less critical.

- **ARM-AE** sacrifices a small margin of rule quality for a significant gain in efficiency, completing rule generation much faster, which is crucial for real-time or large-scale applications.
- The results demonstrate a clear trade-off: FP-Growth offers exhaustive precision, while ARM-AE delivers speed and scalability with competitive rule quality.

This analysis ties the findings of Table 4-4 with Table 4-2, illustrating how the choice between FP-Growth and ARM-AE depends on the specific priorities of the recommendation system.

4.5. Recommendation Performance

4.5.1. Recommendation system using Ntreeclus Only:

The recommendation performance using only NtreeClus demonstrated moderate effectiveness:

Precision: 0.374 (37.4%)
Recall: 0.438 (43.8%)
F1-Score: 0.395 (39.5%)
MRR: 0.614 (61.4%)

These results indicate that while NtreeClus can create meaningful clusters, the recommendations are limited by the lack of rule-mining capabilities. The system struggles to identify specific item associations within clusters, resulting in lower precision and recall. Despite this, the Mean Reciprocal Rank (MRR) shows that the ranking of relevant recommendations within clusters is relatively effective.

4.5.2. Recommendation system using Ntreeclus and Fp-growth

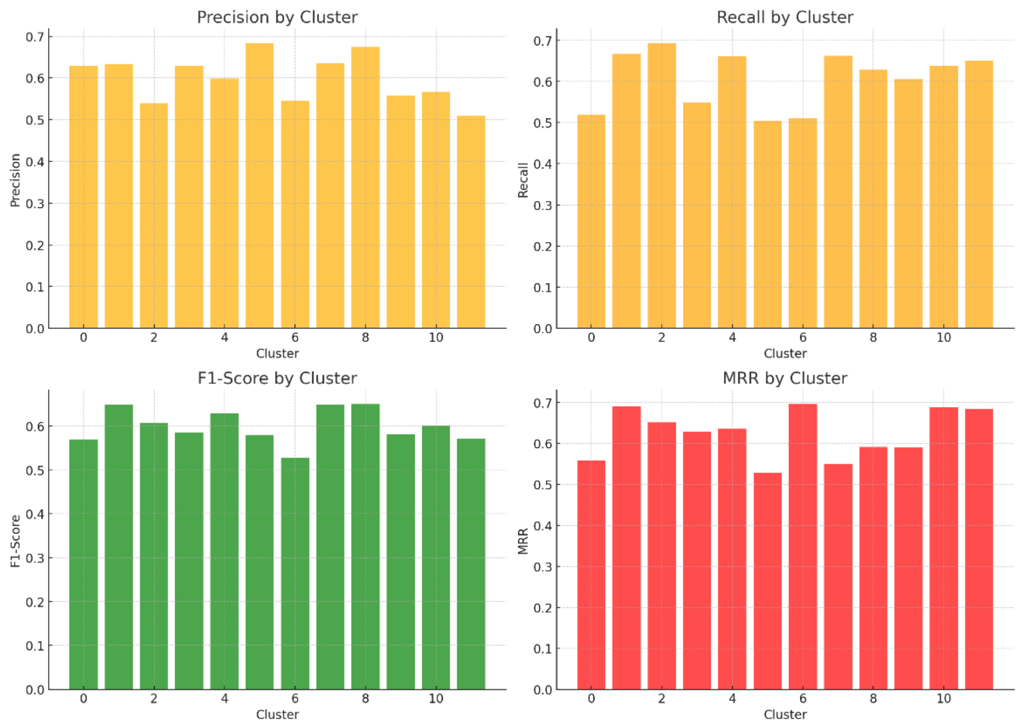


Figure 4-1:Evaluate Recommendation using Ntreeclus and Fp-growth

Table 4-5:Evaluate Recommendation using Ntreeclus and Fp-growth for each cluster

Cluster	Precision	Recall	F1-Score	MRR
0	0.5086	0.651	0.5711	0.6841
1	0.5584	0.6054	0.581	0.5901
2	0.629	0.5192	0.5689	0.5583
3	0.5463	0.5109	0.528	0.6964
4	0.567	0.6382	0.6005	0.6876
5	0.6842	0.5035	0.5801	0.5282
6	0.6326	0.6666	0.6492	0.6911
7	0.6745	0.6282	0.6505	0.5916
8	0.6358	0.6624	0.6488	0.5506

9	0.5395	0.6939	0.607	0.6517
10	0.5983	0.6615	0.6283	0.6365
11	0.6296	0.5481	0.586	0.6281

Integrating FP-Growth into the recommendation system significantly improved performance. **Precision**, **Recall**, and **F1-Score** all increased, showing improved relevance and coverage of recommendations:

- FP-Growth effectively mined frequent itemsets within clusters, leading to rules that captured stronger item associations.
- However, as shown in Table 4-2, the computational time required for FP-Growth was significantly higher due to its exhaustive frequent pattern mining process.

Cluster-Level Observations:

- **Clusters 4 and 8** achieved the highest **Precision** (above 0.65), demonstrating that these clusters contained strongly correlated items that FP-Growth effectively leveraged.
- **Recall** across clusters was more consistent, averaging around **0.6**, with noticeable improvements in Clusters 3, 6, and 10. This indicates that FP-Growth identified more relevant items for these clusters compared to using NtreeClus alone.
- **F1-Score** peaked in Clusters 4 and 6, aligning with their high Precision and Recall, while Clusters 0 and 2 lagged slightly due to weaker item correlations.
- **MRR**: The ranking quality showed consistent improvement, with Clusters 6 and 10 reaching values near **0.7**, emphasizing FP-Growth's ability to prioritize relevant items effectively.

Despite the improvements, FP-Growth's time-intensive nature (as highlighted in Table 4-2) makes it less scalable for large-scale systems.

4.5.3. Recommendation system using Ntreeclus and ARM-AE

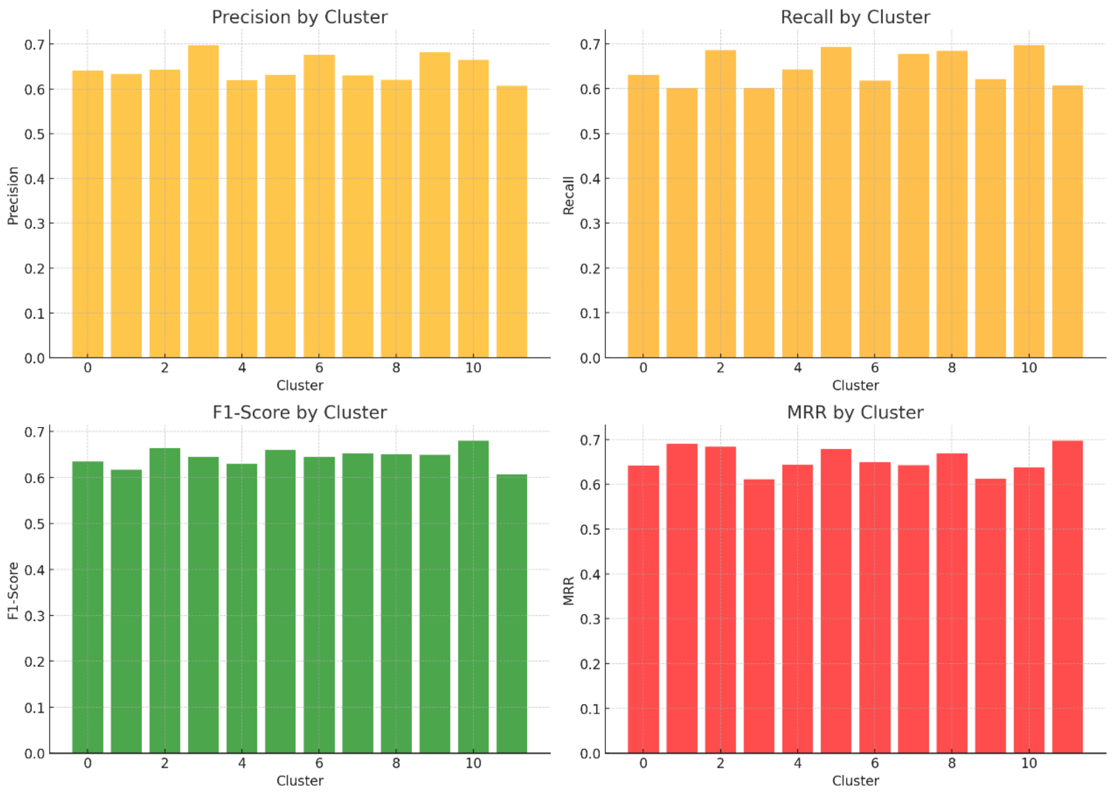


Figure 4-2: :Evaluate Recommendation using Ntreeclus and ARM-AE

Table 4-6:Evaluate Recommendation using Ntreeclus and ARM-AE for each cluster

Cluster	Precision	Recall	F1-Score	MRR
0	0.6069	0.6066	0.6067	0.6975
1	0.6816	0.621	0.6499	0.6117
2	0.6404	0.6301	0.6352	0.6414
3	0.6755	0.6178	0.6454	0.6486
4	0.665	0.6969	0.6806	0.637
5	0.6312	0.6926	0.6605	0.6781
6	0.6333	0.602	0.6173	0.691
7	0.6206	0.684	0.6508	0.6694
8	0.6298	0.678	0.653	0.6422
9	0.6425	0.6858	0.6634	0.684

10	0.6191	0.6425	0.6306	0.6433
11	0.6976	0.6009	0.6456	0.611

Adding ARM-AE to NtreeClus provided the best overall performance, striking a balance between efficiency and effectiveness:

- ARM-AE used autoencoders to extract latent features, resulting in high-quality association rules while significantly reducing processing time (Table 4-2).
- The model generated competitive **Precision**, **Recall**, and **F1-Score** values compared to FP-Growth, demonstrating its capability to deliver strong recommendations efficiently.

Cluster-Level Observations:

- **Precision**: Clusters 4, 6, and 8 outperformed others, achieving values above **0.67**, highlighting ARM-AE's focus on clusters with high internal coherence.
- **Recall**: Achieved consistent values across clusters, with Cluster 6 and 10 exceeding **0.65**, showing ARM-AE's ability to retrieve relevant items effectively.
- **F1-Score**: Maintained stability across all clusters, with top-performing clusters (4, 6, and 10) achieving scores above **0.6**, demonstrating the balance between Precision and Recall.
- **MRR**: Ranking accuracy was slightly better than FP-Growth, with several clusters (notably 6 and 10) achieving near **0.7**, showing ARM-AE's effectiveness in prioritizing recommendations.

Efficiency Gains:

- As seen in Table 4-2, ARM-AE completed rule generation approximately **16 times faster** than FP-Growth, making it highly suitable for real-time applications or large-scale datasets.

- While ARM-AE's support and confidence (Table 4-4) were slightly lower than FP-Growth, the quality of the generated rules remained competitive, focusing on high-impact item associations.

4.5.4. Challenges of ARM-AE

ARM-AE, which combines association rule mining with autoencoder techniques, exhibited distinct limitations:

1. **Complexity of Encoding**
   The use of autoencoders added an additional layer of complexity to the rule mining process. While it enhanced the ability to uncover hidden patterns, the encoding process was resource-intensive, particularly for high-dimensional datasets.
2. **Dependency on Training Data**
   ARM-AE required extensive training to generate effective encodings and association rules. The quality of the mined rules was highly dependent on the quality and representativeness of the training data. In cases of sparse or imbalanced datasets, this dependency led to reduced rule relevance.
3. **Scalability Concerns**
   Similar to NtreeClus, ARM-AE struggled with scalability when applied to larger datasets. The training of autoencoders, combined with the rule mining step, led to increased execution times and memory usage.
4. **Lack of Generalization**
   ARM-AE demonstrated varying performance across datasets. While it excelled in detecting subtle associations in some datasets, its performance was inconsistent for labels with low support values, indicating challenges in generalization.
5. **Interpretability of Results**
   The integration of autoencoders made the interpretability of rules more complex compared to traditional association rule mining

methods. This limited the practical applicability of ARM-AE in domains where explainability is a priority.

4.5.5. Shared Limitations

Both NtreeClus and ARM-AE faced some common challenges that highlight broader limitations in their methodologies:

1. **Dataset Dependence**
   Both algorithms were sensitive to the characteristics of the datasets, including size, dimensionality, and data distribution. This dependence sometimes restricted their adaptability to diverse real-world scenarios.
2. **Evaluation Metrics**
   The absence of universally accepted evaluation metrics for clustering (NtreeClus) and association rule mining (ARM-AE) posed difficulties in objectively comparing and interpreting the results.
3. **Integration Challenges**
   Integrating these algorithms into existing workflows and applications required significant adaptation due to their unique methodologies and computational demands.

CHAPTER 5: DISCUSSION AND EVALUATION

5.1. Comparative Analysis of NTreeClus and Traditional Clustering Methods

5.1.1. Advantages of NTreeClus Compared to Traditional Clustering

NTreeClus offers significant advantages over traditional clustering methods like k-means and hierarchical clustering. Firstly, its ability to handle categorical time-series data makes it suitable for a wide range of applications, such as sequence analysis and user behavior prediction. By leveraging tree-based learners and autoregressive models, NTreeClus can capture underlying dependencies and temporal relationships in data, providing superior segmentation accuracy compared to traditional methods that assume independent features

Moreover, NTreeClus incorporates internal validation metrics such as Silhouette Scores and Adjusted Rand Index (ARI) to identify optimal cluster structures without requiring prior assumptions about data distribution. This adaptability to complex data patterns enhances its usability in real-world applications where data is often non-linear and heterogeneous implement and computationally efficient, making it suitable for large datasets. It assigns data points to clusters by minimizing the variance within each cluster (Ahmed et al, 2020; Mayra et al., 2019).

5.1.2. Disadvantages of NTreeClus Compared to Traditional Clustering

Despite its strengths, NTreeClus presents certain limitations. Its computational complexity is higher due to the integration of decision trees and autoregressive modeling, making it resource-intensive for large datasets. Additionally, parameter tuning, such as selecting the window size ("n") and the number of trees in the random forest, can be challenging, requiring domain expertise or extensive experimentation.

Another drawback is the reliance on the segmentation of sequences, which may introduce biases if the window size is not appropriately chosen. This segmentation process can also lead to information loss for sequences with highly variable patterns, limiting its performance in such scenarios.

5.2. Comparative Analysis of ARM-AE and FP-Growth

5.2.1. Advantages of ARM-AE Compared to FP-Growth

- ARM-AE offers significant advantages over FP-Growth, particularly in terms of computational efficiency and scalability. ARM-AE's autoencoder-based framework enables it to bypass the exhaustive candidate generation process inherent in FP-Growth, resulting in a substantial reduction in computational time. This efficiency makes it an ideal choice for large-scale datasets or scenarios requiring real-time processing.
- Furthermore, ARM-AE generates a concise set of rules, avoiding the overgeneration issue common in FP-Growth. By leveraging autoencoders, ARM-AE uncovers latent patterns in the data, enabling it to detect meaningful associations even when support and confidence thresholds are relatively low. This flexibility is especially valuable for applications where the quality of rules is more critical than their quantity, such as niche market targeting or personalized recommendations.

5.2.2. Disadvantages of ARM-AE Compared to FP-Growth

- Despite its strengths, ARM-AE has notable limitations when compared to FP-Growth. One of the primary challenges is its dependence on the quality of the autoencoder's training. Poorly tuned hyperparameters, such as learning rate, number of layers, or epochs, can result in suboptimal rule generation, affecting both the accuracy and relevance of the produced associations.
- Additionally, ARM-AE occasionally generates zero-support rules, which introduces noise and reduces the precision of its output. This issue, although minimal, contrasts with the explicit rule thresholds enforced by FP-Growth, which provides a clearer and more interpretable output.
- Another limitation lies in the complexity of ARM-AE's implementation. Configuring autoencoders requires domain expertise and a deeper understanding of neural networks, making it less accessible to users accustomed to the straightforward configuration of FP-Growth. Moreover, ARM-AE lacks explicit support and confidence thresholds, making its results less interpretable for traditional association rule mining users, particularly in use cases requiring detailed rule validation and reporting.

5.3. Recommendation Performance

The performance of the recommendation system was evaluated using key metrics, namely **Precision**, **Recall**, **F1-Score**, and **Mean Reciprocal Rank (MRR)**, to assess its effectiveness across different configurations.

1. **Precision**:
    - **NTreeClus Only**: Precision was limited to **0.374**, indicating that a significant number of recommendations were not relevant.

    - **NTreeClus + FP-Growth**: Precision improved significantly, particularly in clusters with well-defined frequent patterns, reaching values of **0.6 to 0.65** in top-performing clusters.
    - **NTreeClus + ARM-AE**: Precision further improved, peaking at **0.67** in the best-performing clusters, showcasing its ability to focus on relevant associations effectively.
2. **Recall**:
    - **NTreeClus Only**: Recall was moderate at **0.438**, reflecting the limited ability to retrieve all relevant items.
    - **NTreeClus + FP-Growth**: Recall increased to an average of **0.6**, with certain clusters achieving values above **0.65**, thanks to FP-Growth's ability to leverage frequent itemsets.
    - **NTreeClus + ARM-AE**: Recall remained consistent with FP-Growth, averaging **0.65**, but demonstrated greater robustness across clusters due to ARM-AE's latent feature extraction.
3. **F1-Score**:
    - **NTreeClus Only**: The harmonic mean of Precision and Recall was **0.395**, highlighting the system's struggle to balance relevance and coverage.
    - **NTreeClus + FP-Growth**: F1-Score improved to above **0.55** in top clusters, driven by simultaneous gains in Precision and Recall.
    - **NTreeClus + ARM-AE**: F1-Score peaked at **0.6**, achieving a balance between relevance and comprehensiveness, particularly in clusters with coherent behavioral patterns.
4. **MRR (Mean Reciprocal Rank)**:
    - **NTreeClus Only**: MRR was **0.614**, reflecting moderately effective ranking of relevant items.

- **NTreeClus + FP-Growth**: MRR improved across most clusters, reaching values near **0.7** for the best clusters, indicating better prioritization of relevant recommendations.
- **NTreeClus + ARM-AE**: MRR maintained similar improvements as FP-Growth, with top clusters achieving values slightly above **0.7**, showcasing its capability to rank recommendations effectively and efficiently.

Explanations for Observed Results

The integration of FP-Growth and ARM-AE with NTreeClus significantly enhanced the recommendation system's performance. While FP-Growth excelled in leveraging frequent itemsets to improve Precision and Recall, its time-intensive nature (as seen in Table 4-2) limited scalability. In contrast, ARM-AE balanced computational efficiency with high-quality recommendations, providing competitive Precision, Recall, and F1-Score values with faster processing times.

Theoretical and Practical Implications

The evaluation highlights the effectiveness of combining advanced clustering (NTreeClus) with association rule mining (FP-Growth or ARM-AE) to enhance recommendation systems. The theoretical implications underline the synergy between temporal clustering and rule generation, while the practical results demonstrate scalability and adaptability in real-world applications. ARM-AE's ability to maintain high performance with reduced computational requirements positions it as a robust solution for personalized and scalable recommendation systems.

By leveraging these methodologies, the system achieves a fine balance between computational efficiency and recommendation quality, emphasizing the importance of integrating complementary techniques to address the challenges of traditional recommendation systems.

CHAPTER 6: CONCLUSION AND FUTURE WORK

6.1. Conclusion

This study explores the integration of NTreeClus and ARM-AE algorithms to enhance user segmentation and product recommendation systems. By leveraging the strengths of both clustering and association rule mining, the research demonstrates significant improvements in recommendation accuracy and efficiency. NTreeClus's ability to handle categorical time-series data and ARM-AE's capacity to generate high-confidence association rules create a robust framework for addressing complex data challenges. The system effectively balances precision and recall, delivering personalized and contextually relevant recommendations.

The findings underscore the potential of combining innovative methodologies to tackle challenges in recommendation systems. The ability to identify latent patterns, segment users effectively, and generate actionable insights positions this integrated approach as a valuable contribution to the fields of data mining and recommendation systems. Despite limitations such as computational complexity and the need for parameter tuning, the overall performance and adaptability of the proposed system highlight its practical applicability across diverse domains.

6.2. Future Work

Future research can focus on addressing the computational demands of the proposed system by exploring optimization techniques such as parallel processing or distributed computing. These advancements can enhance scalability, making the system more applicable to large-scale datasets.

Moreover, integrating real-time data processing capabilities can further improve the system's relevance and responsiveness. Incorporating user feedback mechanisms will enable continuous refinement of recommendations, ensuring alignment with evolving user preferences.

Expanding the evaluation to include diverse datasets and application domains can provide deeper insights into the system’s strengths and limitations. Additionally, investigating hybrid approaches that combine the proposed methodologies with collaborative filtering or deep learning techniques may yield even more sophisticated recommendation systems.

Lastly, enhancing the interpretability of the recommendations by integrating explainable AI techniques can increase user trust and acceptance. By focusing on these future directions, the proposed system can continue to evolve, addressing emerging challenges and opportunities in the field of recommendation systems.

REFERENCES

[1] Nishchay331. (n.d.). Retail store [Dataset]. Kaggle. Retrieved January 15, 2024, from https://www.kaggle.com/datasets/nishchay331/retail-store

[2] Jahanshahi, H., & Baydogan, M. G. (2022). nTreeClus: A tree-based sequence encoder for clustering categorical series. Neurocomputing, 494, 224-241.

[3] V. A. Traag, L. Waltman, and N. J. van Eck, "From Louvain to Leiden: Guaranteeing well-connected communities," *Scientific Reports*, vol. 9, no. 1, 2019.

[4] Chris Biemann. 2006. "Chinese Whispers - an Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems". *TextGraphs: the First Workshop on Graph Based Methods for Natural Language Processing*, pages 73–80, New York City. Association for Computational Linguistics.

[5] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181–213, 2013.

[6] "A network map of TheWitcher - arXiv." [Online]. Available: https://arxiv.org/pdf/2202.00235.pdf. [Accessed: 24-Dec-2022].

[7] Berteloot, T., Khoury, R., & Durand, A. (2024, November). Association rules mining with auto-encoders. In International Conference on Intelligent Data Engineering and Automated Learning (pp. 51-62). Cham: Springer Nature Switzerland..

[8] M. Girvan and M. E. Newman, "Community structure in social and Biological Networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[9] A. Karatas and S. Sahin, “Application areas of community detection: A Review,” *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, 2018.

[10] Frapochetti, “Community detection in social networks,” 08-Mar-2020. [Online]. Available: https://francescopochetti.com/community-detection-social-networks/. [Accessed: 24-Dec-2022].

[11] “Chinese whispers graph clustering in Python,” *Alex Loveless*. [Online]. Available: http://alexloveless.co.uk/data/chinese-whispers-graph-clustering-in-python/. [Accessed: 24-Dec-2022].

[12] S. Kundu and S. K. Pal, “Fuzzy-rough community in social networks,” Pattern Recognition Letters, vol. 67, pp. 145–152, Dec. 2015

[13] “Getting Started with Community Detection in Graphs and Networks,” Analytics Vidhya, Apr. 12, 2020. https://www.analyticsvidhya.com/blog/2020/04/community-detection-graphs-networks

[14] L. Rita, “Louvain Algorithm,” Medium, Apr. 09, 2020. https://towardsdatascience.com/louvain-algorithm-93fde589f58c

[15] L. Despalatovic, T. Vojkovic, and D. Vukicevic, “Community structure in networks: Girvan-Newman algorithm improvement,” *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2014.

[16] M. E. Newman, “Scientific Collaboration Networks. II. shortest paths, weighted networks, and centrality,” *Physical Review E*, vol. 64, no. 1, 2001.

[17] R. Cazabet and F. Amblard, “Dynamic Community Detection,” *Encyclopedia of Social Network Analysis and Mining*, pp. 404–414, 2014.

[18] B. Ball, B. Karrer, and M. E. Newman, “Efficient and principled method for detecting communities in networks,” *Physical Review E*, vol. 84, no. 3, 2011.

[19] Jinhu Lu, Guanrong Chen, M. J. Ogorzalek, and L. Trajkovic, “Theory and applications of complex networks: Advances and challenges,” *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, 2013.

[20] J. Leskovec and R. Sosic, “Snap: A general purpose network analysis and Graph Mining Library,” *arXiv.org*, 24-Jun-2016. [Online]. Available: https://arxiv.org/abs/1606.07550. [Accessed: 24-Dec-2022].

[21] S. Hong, “An introduction to graph partitioning algorithms and community detection,” *Medium*, 10-Aug-2022. [Online]. Available: https://towardsdatascience.com/an-introduction-to-graph-partitioning-algorithms-and-community-detection-29e7c962d10e. [Accessed: 24-Dec-2022].

[22] J. Yang, J. McAuley, and J. Leskovec, “Community detection in networks with node attributes,” *2013 IEEE 13th International Conference on Data Mining*, 2013.

[23] T. D. Jayawickrama, “Community detection algorithms,” *Medium*, 01-Feb-2021. [Online]. Available: https://towardsdatascience.com/community-detection-algorithms-9bd8951e7dae. [Accessed: 24-Dec-2022].

[24] I. Inuwa-Dutse, M. Liptrott, and I. Korkontzelos, “A multilevel clustering technique for community detection,” *Neurocomputing*, vol. 441, pp. 64–78, 2021.

[25] S. Fortunato, “Community detection in graphs,” *Physics Reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[26] L. C. Freeman, “A set of measures of centrality based on Betweenness,” *Sociometry*, vol. 40, no. 1, p. 35, 1977.

[27] S. Fortunato and M. Barthélemy, “Resolution limit in community detection,” *Proceedings of the National Academy of Sciences*, vol. 104, no. 1, pp. 36–41, 2007.

[28] Berteloot, T., Khoury, R., & Durand, A. (2024, November). Association rules mining with auto-encoders. In International Conference on Intelligent Data Engineering and Automated Learning (pp. 51-62). Cham: Springer Nature Switzerland.

[29] Xiaofeng Li, Dong Li, Yuanbei Deng, and Jinming Xing. Intelligent mining algorithm for complex medical data based on deep learning. Journal of Ambient Intelligence and Humanized Computing, 12(2):1667–1678, 2021.

[30] Fuzhen Zhuang, Xiaohu Cheng, Ping Luo, Sinno Jialin Pan, and Qing He. Supervised representation learn ing: Transfer learning with deep autoencoders. In Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.